D^{2}PLAN: Dual-Agent Dynamic Global Planning for Complex Retrieval-Augmented Reasoning

Anonymous ACL submission

Abstract

Recent search-augmented LLMs trained with reinforcement learning (RL) can interleave searching and reasoning for multi-hop reasoning tasks. However, they face two critical failure modes as the accumulating context becomes flooded with both crucial evidence and irrelevant information: (1) deficient search chains that contain incorrect queries or omit retrieval of critical information, and (2) vulnerability to retrieval noise that causes models to misidentify distractors as valid evidence. To address these challenges, we propose D^{2}PLAN, a **D**ual-agent **D**ynamic global **Plan**ning paradigm for complex retrieval-augmented reasoning. D^{2}PLAN operates through the collaboration of a *Reasoner* and a *Purifier*: the *Reasoner* constructs explicit global plans during reasoning and dynamically adapts them based on retrieval feedback; the *Purifier* assesses retrieval relevance and condenses key information for the *Reasoner*. We further introduce a two-stage training framework consisting of supervised fine-tuning (SFT) cold-start on synthesized trajectories and RL with plan-oriented rewards to teach LLMs to master the D^{2}PLAN paradigm. Extensive experiments demonstrate that D^{2}PLAN enables more coherent multi-step reasoning and stronger resilience to irrelevant information, thereby achieving superior performance on challenging QA benchmarks.

1 Introduction

Recent advancements in reinforcement learning (RL) have empowered Large Language Models (LLMs) with the capability to perform autonomous retrieval during reasoning (Song et al., 2025; Jin et al., 2025b; Chen et al., 2025; Sun et al., 2025; Zheng et al., 2025; Zhang et al., 2025). Transcending the traditional Retrieval-Augmented Generation (RAG) paradigm (Lewis et al., 2020; Gao et al., 2023), these models are no longer passive consumers of retrieved documents. Instead, they actively determine *when*, *what*, and *how* to search, subsequently integrating the retrieved information into their reasoning processes (Lin et al., 2025), thereby enhancing performance on knowledge-intensive multi-hop reasoning tasks.

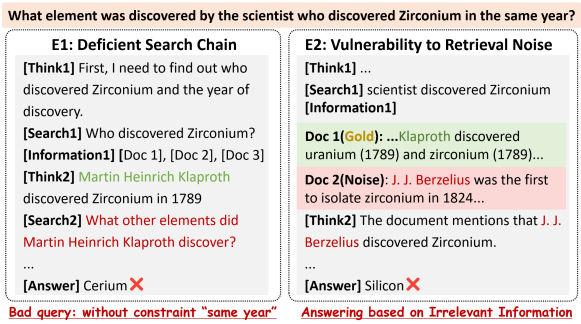

Figure 1: Illustrative examples of 2 typical error types.

However, we observe that as LLMs iteratively invoke retrievers to gather evidence for complex questions, the context becomes inundated with documents retrieved at various steps. This lengthy context, containing both key evidence and substantial irrelevant information, dilutes the model's attention to the cues in the original question and retrieved critical evidence (Hsieh et al., 2024; Fang et al., 2025). As shown in Figure 1 (Left), the model loses track of temporal constraints in the original question, leading to a newly constructed query that lacks global context and causes a drift in retrieval targets. In other scenarios (Figure 1, Right), when faced with a lengthy context flooded with all retrieved information, the model struggles to discriminate between valid evidence and distractors, leading to erroneous conclusions derived from irrelevant information. We categorize these pervasive failure modes into two primary types: **(E1) Deficient Search Chain:** The query sequences iteratively generated by models often contain incorrect queries, causing divergence from the original reasoning objective, or lack necessary queries, omitting retrieval of essential knowledge points. **(E2)**

Vulnerability to Retrieval Noise: Models tend to misidentify irrelevant information in retrieved results as valid evidence, and fail to re-retrieve when retrieval yields entirely irrelevant results.

After analyzing the results of state-of-the-art methods, we find that both E1 and E2 are prevalent and account for 29% and 63% of total failures, respectively (§4.6). These phenomena suggest that to better deal with complex multi-hop reasoning tasks, models need precise perception and control over the retrieval-augmented reasoning process. Specifically, two core capabilities are essential: **(C1) Dynamic Global Planning:** The model must be capable of formulating an explicit global plan and maintaining awareness of it throughout the reasoning process. Furthermore, it should be able to dynamically adapt this plan based on retrieval feedback, including revising sub-questions and rewriting queries. This ability prevents the drift in retrieval targets and the omission of critical reasoning or retrieval steps, while enabling recovery from retrieval failures. **(C2) Robust Noise Resistance:** The model should be able accurately distinguish key evidence from irrelevant information in retrieved results. This capability ensures reasoning is grounded in reliable evidence and enables the model to recognize retrieval failures, thereby performing self-correction by triggering plan adaptation in C1.

To equip models with such capabilities, we propose D^{2}PLAN, a **D**ual-agent **D**ynamic global **Plan**ning paradigm supported by a two-stage training framework. D^{2}PLAN operates through the collaboration of two agents: a *Reasoner* and a *Purifier*. The *Reasoner* performs retrieval-augmented reasoning with explicit dynamic global planning, while the *Purifier* evaluates retrieval relevance and distills key information for the *Reasoner*. For a given question, the *Reasoner* autonomously constructs a global plan (i.e., an ordered sequence of sub-questions), sequentially addresses each sub-question, and dynamically adapts the plan based on feedback from the *Purifier*.

The training process of D^{2}PLAN consists of two phases: SFT Cold-Start and SPLANRL. In the first phase, we design a pipeline to manually implement the workflow of D^{2}PLAN and synthesize data using a powerful teacher model to fine-tune both the *Reasoner* and *Purifier*. This cold-start stage enables the *Reasoner* to acquire preliminary reasoning capabilities with dynamic global planning, and enables the *Purifier* to learn accurate relevance assessment and information condensation. In the second phase, we introduce SPLANRL, which utilizes plan-oriented rewards to enhance the *Reasoner*'s reasoning capability by guiding it to correctly formulate and adapt global plans. To validate the effectiveness of D^{2}PLAN, we conducted extensive experiments on challenging QA benchmarks. Notably, our method achieves an average absolute performance improvement of 3.8% over the state-of-the-art baseline across six datasets.

Our contributions are as follows: (1) We identify two prevalent failure modes of existing LLMs when handling lengthy retrieval-augmented reasoning contexts: Deficient Search Chain and Vulnerability to Retrieval Noise. (2) We propose D^{2}PLAN, a dual-agent dynamic planning paradigm supported by a two-stage training framework, to enhance the global planning and noise resistance capabilities of search-augmented LLMs, thereby mitigating these failure modes. (3) Extensive experiments demonstrate that D^{2}PLAN enables more coherent query formulation and stronger resilience to retrieval noise during reasoning, achieving superior performance on challenging QA benchmarks.

2 Methodology

In this section, we present our D^{2}PLAN paradigm and the corresponding training framework. First, we provide an overview of the D^{2}PLAN workflow[1], introducing how its dual-agent architecture and dynamic global planning mechanism operate when solving challenging problems (§2.1). Then, we detail how we teach LLMs to master this paradigm through a two-stage training framework comprising SFT cold-start (§2.2) and SPLANRL (§2.3).

2.1 Overview of D^{2}PLAN

Dual-Agent Architecture As shown in Figure 3, D^{2}PLAN comprises two agents: a *Reasoner* and a *Purifier*. The *Reasoner* is responsible for executing core planning, reasoning, and invoking search tools. When invoking search tools, the *Reasoner* generates a query and specifies the target sub-question to be solved. A retriever returns candidate documents based on the query. These documents are not directly passed to the *Reasoner*, but first analyzed by the *Purifier*. The *Purifier* focuses on assessing whether the candidate documents contain key information for solving the target sub-question. If relevant information exists, the *Purifier* extracts it

[1] See Appendix A for pseudocode of the workflow.

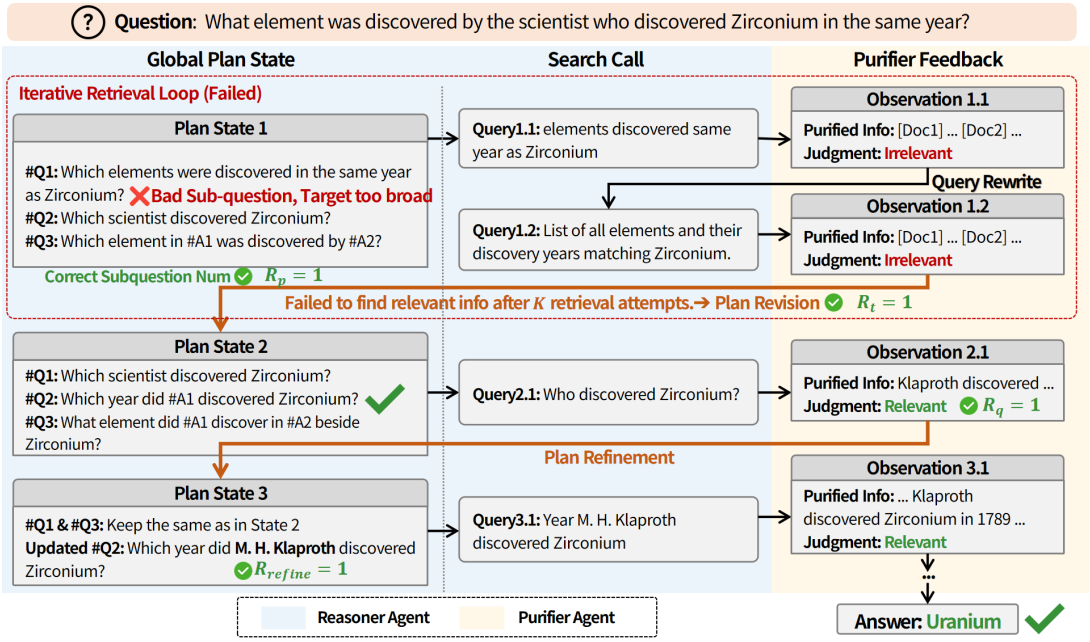

Figure 2: An illustrative example of our D^{2}PLAN paradigm resolving a multi-hop reasoning question, where the correct number of sub-questions in the initial plan and effective plan adaptation are rewarded in the RL phase.

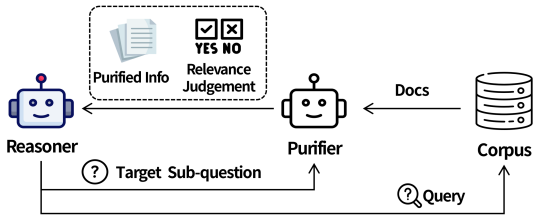

Figure 3: Overview of the dual-agent architecture.

precisely; otherwise, it provides a brief summary of the retrieved documents. The purified information and relevance judgment are then fed back to the *Reasoner*, enabling it to adapt the global plan and continue reasoning.

Dynamic Global Planning As shown in Figure 2, given an input question, the *Reasoner* autonomously assesses its complexity and constructs a global problem-solving plan. Single-hop questions require no decomposition, whereas multi-hop questions are decomposed into an ordered sequence of sub-questions to form an initial plan. In this sequence, each sub-question depends only on preceding ones and can reference their answers using placeholders like `#A_j` (denoting the answer to the j-th sub-question), which allows the *Reasoner* to solve them sequentially. When solving each sub-question, the *Reasoner* can iteratively generate queries and invoke search tools based on feedback from the *Purifier*, continuing until relevant evidence is obtained or the maximum number of attempts K is reached. Upon successfully solving a sub-question, the *Reasoner* refines the subsequent sub-question based on the resolved context, making it more specific and self-contained to facilitate effective retrieval and prevent divergence from the original reasoning objective. Conversely, if a sub-question remains unsolved after reaching the maximum retrieval attempts, the *Reasoner* revises the remaining sub-questions, including the current unsolved one, to explore alternative problem-solving paths for self-correction.

2.2 SFT Cold-Start

We use SFT to train the dual agents, guiding *Reasoner* to acquire preliminary retrieval-augmented reasoning capabilities with dynamic global planning, and enabling *Purifier* to master relevance assessment and information condensation.

2.2.1 Synthesizing Trajectories

To obtain high-quality SFT training data, we design a pipeline that guides a powerful teacher LLM to play the roles of *Reasoner* and *Purifier*.[2] Specifically, we employ a series of curated prompts (see Appendix J) to step-by-step guide the teacher model to simulate the workflow described in §2.1, executing a series of operations including question decomposition, query rewriting, plan refinement, and plan revision, etc. Through this process, we

[2]See Appendix B for detailed algorithms and pseudocode.

obtain a set of reasoning trajectories synthesized according to the D^{2}PLAN workflow. Each trajectory is also accompanied by the corresponding input-output pairs of the *Purifier* for relevance judgment and information purification at each retrieval step.

2.2.2 Constructing SFT Data

To construct high-quality SFT training data, we first filter the synthesized reasoning trajectories based on two criteria: (1) the final answer exactly matches the gold answer; (2) the trajectory contains at least one retrieval call. The second criterion ensures that the model prioritizes learning to leverage retrieval tools for problem-solving rather than relying on potentially erroneous internal parametric knowledge. Based on this filtered set of high-quality trajectories, we construct corresponding training data for both agents. For the *Reasoner*, we further use the teacher LLM to reformat the filtered reasoning trajectories, integrating the segmented reasoning steps synthesized via multiple prompts into a coherent and fluent reasoning chain that serves as the training target. For the *Purifier*, we extract the input-output pairs corresponding to each retrieval step from the filtered reasoning trajectories as its training data.

2.3 SPLANRL

After the SFT stage, we freeze the *Purifier* and introduce SPLANRL to further enhance the *Reasoner*'s reasoning capability by strengthening its dynamic planning proficiency. In addition to the commonly used format and answer rewards in general RL, we design plan-oriented rewards to effectively guide the model's dynamic planning behavior, incentivizing both the quality of plan construction and the effectiveness of plan adjustments.

Initial Planning Reward This reward incentivizes the model to identify the required number of reasoning steps and construct high-quality initial plans. Specifically, for single-hop questions, we assign $R_p = 1$ if the model does not perform question decomposition, and $R_p = 0$ otherwise. For multi-hop questions, we assign $R_p = 1$ if the model performs question decomposition and the number of generated sub-questions matches the hop count annotated in the dataset, and $R_p = 0$ otherwise. We use count-based rather than semantic constraints for two reasons. First, count-based constraints do not require annotations of sub-questions and sub-answers, making them more practical to implement. Second, count-based constraints effectively prevent the model from generating redundant sub-questions or missing necessary ones, thereby improving the conciseness and effectiveness of planning (§4.4).

Plan Adaptation Reward This reward aims to reinforce the model's capability of dynamically adjusting plans throughout the reasoning process, encompassing two complementary mechanisms: *plan refinement* and *plan revision*.

Plan Refinement Reward. For each sub-question that depends on the preceding ones, we use the Plan Refinement Reward R_{refine} to encourage the model to refine it after solving those dependencies—filling in placeholders and making subsequent sub-questions more specific and self-contained. A successful refinement requires that the refined sub-question exists in the reasoning trajectory and contains no placeholders, which we verify using regular expression matching. If all such sub-questions are successfully refined, the reward $R_{refine} = 1$, and $R_{refine} = 0$ otherwise. This reward not only encourages the model to improve sub-question quality, but also effectively prevents the drift in retrieval targets during multi-hop reasoning.

Plan Revision Reward. We design a revision reward to encourage the model to globally revise the remaining sub-questions when iterative retrieval reaches the maximum number of attempts without finding relevant information. Such failures are often caused by poorly formulated sub-questions, such as those that are overly vague or lack necessary constraints. This reward is jointly determined by timing reward R_t and quality reward R_q. $R_t = 1$ if the model triggers revision only after retrieval reaches the maximum number of failures, preventing premature abandonment; otherwise $R_t = 0$. $R_q = 1$ if the first revised sub-question (i.e., the current unsolved sub-question after revision) successfully retrieves relevant information; otherwise $R_q = 0$. We use the relevance judgment from the *Purifier* to determine whether the retrieval is successful. Specifically, let $c \in \{0, 1\}$ denote answer correctness and $r \in \{0, 1\}$ indicate whether revision was triggered, then:

$$R_{revise} = \begin{cases} 1 & \text{if } c = 1 \\ \lambda \cdot (R_t + R_q) & \text{if } c = 0 \wedge r = 1 \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

By introducing a discount factor λ, R_{revise} provides partial process rewards for trajectories that answer incorrectly but attempt revision. This design effectively guides the model to revise its plan in the

right direction and acquire the "failure-reflection-correction" reasoning pattern.

Combining the two terms, our Plan Adaptation Reward is defined as $R_a = R_{refine} + R_{revise}$.

Answer Reward Following recent work (Zheng et al., 2025; Zhang et al., 2025), we adopt the word-level F1 score between the predicted answer and the ground truth to measure answer correctness. The answer reward is defined as $R_{ans} = \text{F1}(a_{pred}, a_{gt})$, where $\text{F1}(a_{pred}, a_{gt})$ denotes the F1 score between the predicted answer a_{pred} and the golden answer a_{gt}.

Format Reward We treat the format reward as a prerequisite constraint, aiming to ensure the model to generate parsable reasoning paths, which is essential for computing the Initial Planning Reward R_p and Plan Adaptation Reward R_a. For example, the model must wrap the initial plan with `<plan></plan>` tags, the revised plan with `<replan></replan>` tags, and the final answer with `<answer></answer>` tags (see Appendix F for detailed constraints). When all constraints are satisfied, $R_f = 1$, otherwise 0.

Total Reward The final total reward function is defined as: $R_{total} = R_f \cdot (\alpha R_p + \beta R_a + R_{ans})$, where α and β are scaling factors that control the contribution of each reward component.

3 Experimental Setup

Datasets and Evaluation Metrics To comprehensively evaluate the capabilities of D^{2}PLAN, we conduct experiments on widely used open-domain QA benchmarks, including Natural Questions (NQ) (Kwiatkowski et al., 2019), SimpleQA (Wei et al., 2024), HotpotQA (Yang et al., 2018), 2WikiMultihopQA (2Wiki) (Ho et al., 2020), MuSiQue (MSQ) (Trivedi et al., 2022b), and FRAMES (Krishna et al., 2025). For evaluation metrics, we employ Exact Match (EM) and LLM-as-a-Judge (LasJ). For LasJ, we utilize `qwen-plus` (Qwen et al., 2025) to assess the correctness of predicted answers (see Appendix J for the prompt).

Baselines We compare D^{2}PLAN with representative RL-based methods: Search-R1 (Jin et al., 2025b), ReSearch (Chen et al., 2025), AutoRefine (Shi et al., 2025), ZeroSearch (Sun et al., 2025), R1-Searcher (Song et al., 2025), and StepSearch (Zheng et al., 2025). We reproduce the inference results of all baseline methods under the same environment to ensure fair comparison.

Implementation Details We employ Qwen2.5-3B/7B-Instruct (Qwen et al., 2025) as the backbone models. For retrieval, we utilize E5 (Wang et al., 2022) as the embedding model and the 2018 Wikipedia dump (Karpukhin et al., 2020) as the retrieval corpus, retrieving the top-5 passages. For SFT cold-start, we use `qwen-max-latest` as the teacher model to synthesize 4,000 training instances for the *Reasoner* and 11,255 for the *Purifier* from the training sets of NQ, HotpotQA, and MuSiQue. For RL training, we sample 29,762 instances from the same training sets and adopt GRPO (Shao et al., 2024) as the learning algorithm, implemented on verl (Sheng et al., 2025). More implementation details are provided in Appendix C.

4 Results and Analysis

In this section, we aim to answer the following Research Questions (RQs): **RQ1**: How effectively does D^{2}PLAN and each of its components enhance performance in retrieval-augmented question answering? (§4.1, §4.2) **RQ2**: Can Dynamic Global Planning improve the quality of query formulation? (§4.3) **RQ3**: Can SPLANRL enable LLMs to generate and adapt higher-quality plans? (§4.4) **RQ4**: Can Dual-Agent Architecture enhance accuracy and efficiency? (§4.5) **RQ5**: Can D^{2}PLAN reduce these errors: (E1) deficient search chains and (E2) vulnerability to retrieval noise? (§4.6)

4.1 Main Results (RQ1)

As shown in Table 1, **D^{2}PLAN consistently outperforms all baselines at both 3B and 7B scales**, demonstrating strong performance and generalization across in-domain and out-of-domain benchmarks. On average, D^{2}PLAN-3B achieves a 1.34% LasJ gain over the strongest baseline AutoRefine, while D^{2}PLAN-7B yields a 3.83% LasJ improvement over strongest baseline R1-Searcher.

Furthermore, **D^{2}PLAN's advantage widens as reasoning complexity increases**, with relative improvements over Search-R1 scaling from 19.2% on the simpler HotpotQA to 58.3% on the more challenging FRAMES benchmark. These results demonstrate the efficacy of our dual-agent dynamic global planning paradigm in managing complex retrieval-augmented reasoning process.

4.2 Ablation Study (RQ1)

To investigate the contributions of each core component to model performance, we conduct ablation studies on D^{2}PLAN-3B and report the results in

Method	NQ†		SimpleQA‡		HotpotQA†		2Wiki‡		MuSiQue†		FRAMES‡		AVG	
	LasJ	EM	LasJ	EM	LasJ	EM	LasJ	EM	LasJ	EM	LasJ	EM	LasJ	EM
Base LLM	*Qwen2.5-3B-Instruct*													
AutoRefine	**57.01**	41.05	**36.20**	24.90	57.41	41.12	48.08	38.05	28.26	19.49	23.79	11.17	41.79	29.30
Search-R1	44.21	28.70	17.70	8.20	33.49	18.07	33.06	20.92	16.01	6.25	18.20	4.85	27.11	14.50
StepSearch	45.60	32.91	28.20	20.40	37.31	26.64	33.33	28.78	13.24	8.52	13.83	6.55	28.59	20.63
ZeroSearch	51.36	38.42	31.50	22.80	40.61	29.44	30.51	26.42	10.30	6.00	10.32	4.00	29.10	21.18
D²PLAN-3B	55.60	**41.30**	34.10	**25.50**	**57.93**	**43.58**	**52.53**	**45.90**	**33.14**	**23.13**	**25.97**	**14.68**	**43.13**	**32.35**
Base LLM	*Qwen2.5-7B-Instruct*													
R1-Searcher	53.43	40.00	34.60	25.60	59.03	44.32	52.04	46.13	29.83	21.93	25.61	14.32	42.38	32.05
ReSearch	53.85	39.81	36.20	27.30	57.62	42.50	47.34	41.01	29.83	21.43	24.27	14.32	41.52	31.06
Search-R1	55.96	**42.11**	34.40	23.10	51.67	37.31	39.36	33.13	21.51	15.27	19.78	10.19	37.11	26.85
StepSearch	53.07	38.64	34.80	24.90	52.78	38.93	49.57	42.89	28.55	20.31	23.18	13.23	40.32	29.82
ZeroSearch	54.02	41.00	33.10	24.10	45.25	33.48	36.70	31.76	14.23	9.39	18.33	10.07	33.61	24.97
D²PLAN-7B	**58.03**	39.78	**36.50**	**27.50**	**61.59**	**46.17**	**56.32**	**50.02**	**33.47**	**24.41**	**31.31**	**18.08**	**46.21**	**34.33**

Table 1: Main results. †/‡ represents in-domain/out-domain datasets.

Method	2Wiki	MSQ	FRAMES
Only SFT			
D²PLAN-3B	48.64	26.23	25.00
w/o Plan&Adapt	41.40	22.37	18.45
w/o Adapt	46.40	23.42	21.24
SFT+RL			
D²PLAN-3B	52.53	33.14	25.97
w/o Plan&Adapt	49.98	28.13	21.84
w/o Adapt	50.37	29.38	23.30
w/o R_p	50.59	30.45	24.88
w/o R_a	51.65	30.16	24.76

Table 2: Ablation study of D²PLAN. We report LasJ.

Method	VRC	IRC	TRC	SAC
Only SFT				
D²PLAN-3B	2.48	1.98	4.45	1.44
w/o Plan&Adapt	1.26	1.02	2.28	1.11
w/o Adapt	1.57	1.08	2.65	1.22

Table 3: Analysis of retrieval calls on MuSiQue. VRC: Valid Retrieval Calls; IRC: Invalid Retrieval Calls; TRC: Total Retrieval Calls; SAC: Sub-Answer Coverage.

Table 2. Here, *Plan* refers to the initial global planning mechanism for decomposing the question into sub-questions, and *Adapt* refers to the plan adaptation mechanism for plan refinement and revision. Detailed implementations of each ablation variant are provided in Appendix E.

Results show that under the *Only SFT* setting, removing the planning components (*w/o Plan&Adapt* or *w/o Adapt*) significantly degrades performance compared to D²PLAN. Despite being trained on reasoning trajectories synthesized from the same question set, the lack of dynamic planning mechanism prevents these variants from effectively organizing reasoning paths for complex multi-hop problems. This confirms that D²PLAN's gains stem from learned dynamic planning capabilities rather than mere knowledge distillation from the teacher model. Comparing *SFT+RL* with *Only SFT*, RL consistently improves all variants; however, even with RL optimization, *w/o Plan&Adapt* still lags behind D²PLAN, demonstrating that the dynamic planning paradigm provides irreplaceable advantages for structuring complex multi-hop reasoning. Furthermore, the performance drop caused by excluding the initial planning reward R_p (*w/o* R_p) or the plan adaptation reward R_a (*w/o* R_a) underscores the necessity of these rewards in guiding the model to internalize robust plan construction and adaptation capabilities.

4.3 Retrieval Quality (RQ2)

To assess the quality of model-generated queries, we count valid and invalid retrieval calls on MuSiQue, as it provides annotated sub-questions and corresponding sub-answers. Specifically, we define a *valid retrieval* as a retrieval call where the returned results contain the sub-answer to any of the sub-questions. We also report *sub-answer coverage*, which measures the average number of sub-answers covered by all retrieval results. As shown in Table 3, *w/o Adapt* yields more valid retrieval calls than *w/o Plan&Adapt* while maintaining a comparable number of invalid retrieval calls, demonstrating that **global planning assists the model in generating higher-quality query sequences by providing explicit sub-question**

Method	Metric	Hop Count		
		2-hop	3-hop	4-hop
D^{2}PLAN-3B	Pred. Hop	2.03	2.70	3.20
	LasJ (%)	44.4	25.4	13.8
w/o R_p	Pred. Hop	2.26	2.98	3.45
	LasJ (%)	39.7	24.1	12.8

Table 4: Impact of Initial Planning Reward R_p on question decomposition and final accuracy.

Method	First VR (%)	LasJ (%)
D^{2}PLAN-3B	47.11	17.63
w/o R_a	44.79	14.49

Table 5: Impact of Plan Adaptation Reward R_a on MuSiQue. First VR: proportion of valid retrieval on the first query after revision .

decomposition that effectively guides each retrieval step. Furthermore, although D^{2}PLAN incurs more retrieval calls than *w/o Adapt* due to plan adaptation, both valid retrieval calls and sub-answer coverage improve substantially, demonstrating that **the dynamic adaptation mechanism effectively enables robust self-correction, allowing the model to recover from retrieval failures and identifying additional key evidence**.

4.4 Effect of SPLANRL (RQ3)

Initial Planning Reward Guides Concise and Focused Decomposition. To assess the impact of Initial Planning Reward R_p, we compare D^{2}PLAN with *w/o* R_p on subsets of MuSiQue with different hop counts (Table 4). Results indicate that *w/o* R_p consistently predicts higher hop counts than D^{2}PLAN, yet yields lower LasJ scores. This suggests that removing R_p induces over-decomposition, causing the model to generate extraneous sub-questions unrelated to the core problem. For instance, given "Where is Ulrich Walter's employer headquartered?", *w/o* R_p decomposes it into sub-questions including "Who is Ulrich Walter?", which, while tangentially related, does not directly contribute to answering the original question. We sample 600 questions and employ `gemini-3-flash-preview` to detect such redundancies (see Appendix J for the prompt), followed by manual verification. D^{2}PLAN exhibits only 5 such cases, compared to 25 for *w/o* R_p. These findings confirm that R_p guides the model toward concise decomposition strategies, thereby enhancing multi-hop reasoning performance.

Purifier	Metric	Hop Count				
		2	3	4	5+	All
w/o Purifier	LasJ	35.16	26.39	20.15	13.04	27.18
	SC	4.22	5.00	5.57	5.77	4.89
	Len	4451	5305	5921	6164	5180
Qwen-7B-SFT	LasJ	36.77	27.78	22.39	20.65	29.49
	SC	4.03	4.84	5.07	4.99	4.59
	Len	1635	1976	2093	2124	1883

Table 6: Impact of Purifier across different hop counts on FRAMES. LasJ: LLM-as-a-Judge (%); SC: Average search calls; Len: Average context length (tokens).

Plan Adaptation Reward Steers Correct Plan Revisions. To investigate the effect of Plan Adaptation Reward R_a, we compare D^{2}PLAN with *w/o* R_a on MuSiQue. We filter samples where plan revision occurred and report the valid retrieval rate (defined in §4.3) of the first generated query after the revision (First VR), along with the final LasJ accuracy (Table 5). Results show that D^{2}PLAN significantly outperforms *w/o* R_a in both First VR and LasJ. This indicates that R_a enables the model to effectively adapt the plan for self-correction when retrieval fails, leading to improved query reformulation and more accurate reasoning outcomes.

4.5 Effect of Dual-Agent Architecture (RQ4)

To validate the effectiveness of the dual-agent architecture, we train a *Reasoner* baseline on trajectories synthesized from the same question set, but with raw retrieval results instead of Purifier-refined outputs. This ensures the *Reasoner* is not biased toward refined inputs, enabling a fair comparison. We then evaluate the impact of adding the *Purifier* (trained in §2.2) on the FRAMES dataset.

Greater Performance Gains on Harder Questions. As shown in Table 6, incorporating the *Purifier* significantly improves performance across different hop counts while substantially reducing context length. The advantage of the *Purifier* is particularly pronounced on questions with 5 or more hops, where LasJ improves by 7.61% and context length decreases by 65.5%. We attribute the substantially worse performance of *w/o Purifier* to the accumulation of redundant information in raw retrieval results as question difficulty increases, which distracts the *Reasoner* from focusing on critical information both within the current retrieval results and from prior reasoning steps. The *Purifier* effectively mitigates this issue by refining retrieval results and assessing their relevance.

Dual-Agent Architecture Improves Inference Ef-

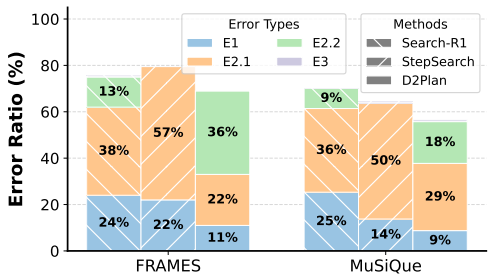

Figure 4: Error type distribution on FRAMES and MuSiQue. Percentages are calculated relative to the total number of questions, including correct predictions.

ficiency. Although the *Purifier* introduces additional inference steps, the end-to-end inference time is reduced. Under the same computational budget, experiments show that using Qwen2.5-7B-SFT as the *Purifier* reduces total inference time from 432s to 381s (∼12% reduction) on the FRAMES dataset. This efficiency gain is attributed to delegating the processing of lengthy retrieval results to the *Purifier*, which substantially reduces the context length for the *Reasoner*, thereby mitigating the quadratic computational cost of self-attention.

4.6 Error Analysis (RQ5)

To understand how D^{2}PLAN mitigates errors, we analyze error type distributions on FRAMES and MuSiQue, comparing it with Search-R1 and StepSearch. We divide **(E2) Vulnerability to Retrieval Noise** into two subtypes: (E2.1) the model reasons with irrelevant retrieval results without attempting remediation; (E2.2) the model recognizes that the initial retrieval result is irrelevant and attempts query rewriting several times but still fails to obtain key evidence, resulting in an incorrect answer. [3] Errors beyond E1 and E2 are categorized as **(E3) Other Errors**. We sample 200 questions each from FRAMES and MuSiQue, using `gemini-3-pro-preview` (Team et al., 2025) to identify the type of the first error in the reasoning trajectory. We manually annotated 100 samples, achieving a Cohen's Kappa of 0.72, confirming Gemini's reliability. As shown in Figure 4, D^{2}PLAN achieves the lowest error rates by substantially reducing E1 and E2.1, validating the efficacy of dynamic global planning and robust noise resistance. The relative rise in E2.2 suggests that as planning and noise handling improve, failures shift toward inherent retrieval limitations and the constrained reasoning capabilities of the 7B model.

[3] E2.2 errors often stem from inherent retrieval system limitations, e.g., the corpus lacks relevant information entirely. See Appendix I for a case study.

5 Related Work

Planning in Retrieval-Augmented Generation. Planning is crucial for RAG systems to handle complex multi-hop questions. Iterative RAG methods (Shao et al., 2023; Yao et al., 2023; Trivedi et al., 2022a; Liu et al., 2024; Jiang et al., 2025; Li et al., 2025) interleave reasoning and retrieval but typically perform local planning, leading to intent drift in lengthy histories. PlanRAG (Lee et al., 2024) introduces a structured approach that generates plans and adjusts them via iterative re-planning. However, it relies on zero-shot prompting of models without parameter optimization.

Reinforcement Learning for Search-Augmented LLMs. Recently, RL has become the dominant paradigm for training search-augmented LLMs (Jin et al., 2025b; Song et al., 2025; Chen et al., 2025; Sun et al., 2025; Shi et al., 2025), as it encourages autonomous exploration and better generalization than SFT (Chu et al., 2025). Further research introduces step-wise process supervision (Zheng et al., 2025; Zhang et al., 2025; Luo et al., 2025a), reward density optimization (Luo et al., 2025b), and self-correction mechanisms (Wang et al., 2025) to enhance fine-grained reasoning and retrieval capabilities. Despite these advances, current methods struggle with noisy and lengthy retrieval-augmented reasoning contexts. Our method addresses these challenges by introducing a dual-agent dynamic global planning paradigm, optimized via a two-stage training framework.

6 Conclusion

We identify two prevalent failure modes of search-augmented LLMs in complex retrieval-augmented reasoning: Deficient Search Chain and Vulnerability to Retrieval Noise. To address these issues, we introduce D^{2}PLAN, a dual-agent dynamic planning paradigm where the Reasoner performs reasoning with explicit dynamic planning and the Purifier evaluates retrieval relevance and distills key information. A two-stage training framework comprising SFT cold-start and planning-oriented RL equips LLMs with the D^{2}PLAN paradigm. Extensive experiments demonstrate that D^{2}PLAN enables more coherent query formulation and stronger resilience to retrieval noise, achieving superior retrieval-augmented reasoning abilities.

7 Limitations

While D^{2}PLAN demonstrates strong performance on open-domain QA benchmarks, several limitations remain. First, although we achieve significant improvements on Qwen2.5-3B and Qwen2.5-7B models, due to computational cost constraints, we are unable to validate our method on larger language models. Second, due to cost limitations, we employ a local dense retriever (E5) with a static Wikipedia corpus rather than more powerful retrieval systems such as commercial search engines (e.g., Google). As revealed in our error analysis, a notable portion of failures (E2.2) may stem from retriever or corpus limitations. Integrating stronger retrieval infrastructure could potentially further enhance the performance of D^{2}PLAN. Third, our approach relies on a powerful teacher model to generate high-quality reasoning trajectories for supervised fine-tuning. This dependency may limit the applicability of our method in scenarios where access to such teacher models is restricted or costly.

References

Mingyang Chen, Tianpeng Li, Haoze Sun, Yijie Zhou, Chenzheng Zhu, Haofen Wang, Jeff Z Pan, Wen Zhang, Huajun Chen, Fan Yang, and 1 others. 2025. Learning to reason with search for llms via reinforcement learning. *arXiv preprint arXiv:2503.19470*.

Tianzhe Chu, Yuexiang Zhai, Jihan Yang, Shengbang Tong, Saining Xie, Dale Schuurmans, Quoc V Le, Sergey Levine, and Yi Ma. 2025. Sft memorizes, rl generalizes: A comparative study of foundation model post-training. *arXiv preprint arXiv:2501.17161*.

Yixiong Fang, Tianran Sun, Yuling Shi, and Xiaodong Gu. 2025. Attentionrag: Attention-guided context pruning in retrieval-augmented generation. *arXiv preprint arXiv:2503.10720*.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yixin Dai, Jiawei Sun, Haofen Wang, and Haofen Wang. 2023. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*, 2(1).

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. 2020. Constructing a multi-hop qa dataset for comprehensive evaluation of reasoning steps. *arXiv preprint arXiv:2011.01060*.

Cheng-Yu Hsieh, Yung-Sung Chuang, Chun-Liang Li, Zifeng Wang, Long Le, Abhishek Kumar, James Glass, Alexander Ratner, Chen-Yu Lee, Ranjay Krishna, and 1 others. 2024. Found in the middle: Calibrating positional attention bias improves long context utilization. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 14982–14995.

Zhouyu Jiang, Mengshu Sun, Lei Liang, and Zhiqiang Zhang. 2025. Retrieve, summarize, plan: Advancing multi-hop question answering with an iterative approach. In *Companion Proceedings of the ACM on Web Conference 2025*, pages 1677–1686.

Bowen Jin, Jinsung Yoon, Priyanka Kargupta, Sercan O Arik, and Jiawei Han. 2025a. An empirical study on reinforcement learning for reasoning-search interleaved llm agents. *arXiv preprint arXiv:2505.15117*.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. 2025b. Search-r1: Training llms to reason and leverage search engines with reinforcement learning. *arXiv preprint arXiv:2503.09516*.

Jiajie Jin, Yutao Zhu, Xinyu Yang, Chenghao Zhang, and Zhicheng Dou. 2024. Flashrag: A modular toolkit for efficient retrieval-augmented generation research. *CoRR*, abs/2405.13576.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick SH Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *EMNLP (1)*, pages 6769–6781.

Satyapriya Krishna, Kalpesh Krishna, Anhad Mohananey, Steven Schwarcz, Adam Stambler, Shyam Upadhyay, and Manaal Faruqui. 2025. Fact, fetch, and reason: A unified evaluation of retrieval-augmented generation. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 4745–4759.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, and 1 others. 2019. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466.

Myeonghwa Lee, Seonho An, and Min-Soo Kim. 2024. Planrag: A plan-then-retrieval augmented generation for generative large language models as decision makers. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 6537–6555.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, and 1 others. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474.

Xiaoxi Li, Guanting Dong, Jiajie Jin, Yuyao Zhang, Yujia Zhou, Yutao Zhu, Peitian Zhang, and Zhicheng Dou. 2025. Search-o1: Agentic search-enhanced large reasoning models. *arXiv preprint arXiv:2501.05366*.

Minhua Lin, Zongyu Wu, Zhichao Xu, Hui Liu, Xianfeng Tang, Qi He, Charu Aggarwal, Xiang Zhang, and Suhang Wang. 2025. A comprehensive survey on reinforcement learning-based agentic search: Foundations, roles, optimizations, evaluations, and applications. *arXiv preprint arXiv:2510.16724*.

Yanming Liu, Xinyue Peng, Xuhong Zhang, Weihao Liu, Jianwei Yin, Jiannan Cao, and Tianyu Du. 2024. Ra-isf: Learning to answer and understand from retrieval augmentation via iterative self-feedback. *arXiv preprint arXiv:2403.06840*.

Jinchang Luo, Mingquan Cheng, Fan Wan, Ni Li, Xiaoling Xia, Shuangshuang Tian, Tingcheng Bian, Haiwei Wang, Haohuan Fu, and Yan Tao. 2025a. Globalrag: Enhancing global reasoning in multi-hop question answering via reinforcement learning. *arXiv preprint arXiv:2510.20548*.

Kun Luo, Hongjin Qian, Zheng Liu, Ziyi Xia, Shitao Xiao, Siqi Bao, Jun Zhao, and Kang Liu. 2025b. Infoflow: Reinforcing search agent via reward density optimization. *arXiv preprint arXiv:2510.26575*.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, and 25 others. 2025. Qwen2.5 technical report. *Preprint*, arXiv:2412.15115.

Zhihong Shao, Yeyun Gong, Yelong Shen, Minlie Huang, Nan Duan, and Weizhu Chen. 2023. Enhancing retrieval-augmented large language models with iterative retrieval-generation synergy. *Preprint*, arXiv:2305.15294.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, and 1 others. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. 2025. Hybridflow: A flexible and efficient rlhf framework. In *Proceedings of the Twentieth European Conference on Computer Systems*, pages 1279–1297.

Yaorui Shi, Shihan Li, Chang Wu, Zhiyuan Liu, Junfeng Fang, Hengxing Cai, An Zhang, and Xiang Wang. 2025. Search and refine during think: Autonomous retrieval-augmented reasoning of llms. *arXiv e-prints*, pages arXiv–2505.

Huatong Song, Jinhao Jiang, Yingqian Min, Jie Chen, Zhipeng Chen, Wayne Xin Zhao, Lei Fang, and Ji-Rong Wen. 2025. R1-searcher: Incentivizing the search capability in llms via reinforcement learning. *arXiv preprint arXiv:2503.05592*.

Hao Sun, Zile Qiao, Jiayan Guo, Xuanbo Fan, Yingyan Hou, Yong Jiang, Pengjun Xie, Yan Zhang, Fei Huang, and Jingren Zhou. 2025. Zerosearch: Incentivize the search capability of llms without searching. *Preprint*, arXiv:2505.04588.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M. Dai, Anja Hauth, Katie Millican, David Silver, Melvin Johnson, Ioannis Antonoglou, Julian Schrittwieser, Amelia Glaese, Jilin Chen, Emily Pitler, Timothy Lillicrap, Angeliki Lazaridou, and 1332 others. 2025. Gemini: A family of highly capable multimodal models. *Preprint*, arXiv:2312.11805.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2022a. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. *arXiv preprint arXiv:2212.10509*.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2022b. Musique: Multi-hop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 10:539–554.

Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. 2022. Text embeddings by weakly-supervised contrastive pre-training. *arXiv preprint arXiv:2212.03533*.

Ziliang Wang, Kang An, Xuhui Zheng, Faqiang Qian, Weikun Zhang, Cijun Ouyang, Jialu Cai, Yuhang Wang, and Yichao Wu. 2025. Erase to improve: Erasable reinforcement learning for search-augmented llms. *arXiv preprint arXiv:2510.00861*.

Jason Wei, Nguyen Karina, Hyung Won Chung, Yunxin Joy Jiao, Spencer Papay, Amelia Glaese, John Schulman, and William Fedus. 2024. Measuring short-form factuality in large language models. *Preprint*, arXiv:2411.04368.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 conference on empirical methods in natural language processing*, pages 2369–2380.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*.

Qi Zhang, Shouqing Yang, Lirong Gao, Hao Chen, Xiaomeng Hu, Jinglei Chen, Jiexiang Wang, Sheng Guo, Bo Zheng, Haobo Wang, and Junbo Zhao. 2025. LeTS: Learning to think-and-search via process-and-outcome reward hybridization. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 5109–5122, Suzhou, China. Association for Computational Linguistics.

Xuhui Zheng, Kang An, Ziliang Wang, Yuhang Wang, and Yichao Wu. 2025. StepSearch: Igniting LLMs search ability via step-wise proximal policy optimization. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 21816–21841, Suzhou, China. Association for Computational Linguistics.

Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, Zheyan Luo, Zhangchi Feng, and Yongqiang Ma. 2024. Llamafactory: Unified efficient fine-tuning of 100+ language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, Bangkok, Thailand. Association for Computational Linguistics.

A D^{2}PLAN Workflow

The inference workflow of D^{2}PLAN is formalized in Algorithm 1. Given an input question Q, the *Reasoner* first assesses its complexity. For single-hop questions, it directly performs iterative retrieval and generates the answer. For multi-hop questions, it decomposes the question into an ordered sequence of sub-questions and solves them sequentially with dynamic plan adaptation.

B Trajectory Synthesis Strategy

Given a question Q, the trajectory synthesis process is formalized in Algorithms 3, 4, and 5.

Iterative Retrieval Loop (Algorithm 3) This component serves as the fundamental building block for retrieving information. It executes a loop for up to K attempts. In each iteration k:

1. **Query Generation**: The teacher model $\mathcal{M}$ generates a reasoning thought r_k and a search query $\mathbf{q}_k$ based on the current history $\mathcal{H}$. When the initial query fails to retrieve relevant information, we employ the query rewriting strategy (Table 16) to refine the search query.

2. **Retrieval and Purification**: The retriever fetches documents $\mathcal{D}_k$, and the *Purifier* (acted by the teacher model, Table 17) extracts information I_k with a relevance judgment J_k.

3. **Termination**: If J_k is `Relevant`, the loop terminates successfully returning the history $\mathcal{H}$. If the loop exhausts K attempts without success, it returns `Failure`.

Single-hop Question Processing (Algorithm 4) The process begins by assessing the question complexity. We denote the reasoning process for this assessment as r_{type}. For single-hop questions, it directly invokes the Iterative Retrieval Loop.

- If the loop succeeds, the teacher model generates the final reasoning r_{final} and answer a based on the collected history (Table 18). The final trajectory includes the complexity assessment, retrieval history, and the final answer.

- If the loop fails, the synthesis process is terminated.

Multi-hop Question Processing (Algorithm 5) The process begins by assessing the question complexity. We denote the reasoning process for this assessment as r_{type}. For multi-hop questions, the process follows the dynamic global planning paradigm:

1. **Initialization**: The model decomposes Q into an initial sub-question plan $\mathcal{P}$ (Table 15).

2. **Execution Loop**: The model iterates through the sub-questions in $\mathcal{P}$. For each sub-question q_i:

   - Before retrieval, we first judge whether external retrieval is necessary (Table 20). If so, it invokes the Iterative Retrieval Loop.
   - **Success (Plan Refinement)**: If retrieval succeeds, the evidence is added to the global context $\mathcal{C}$. The model answers the sub-question (Table 18) and then refines the next sub-question based on $\mathcal{C}$ (Table 19), making it more specific and self-contained.
   - **Failure (Plan Revision)**: If retrieval fails, the model checks if the maximum revision count M has been reached. If not, it triggers plan revision (Table 21) to revise the remaining sub-questions based on the current context and failure information. The execution resumes with the revised plan.

Algorithm 1 D^{2}PLAN Inference Workflow

Input: Question Q, Reasoner $\mathcal{R}$, Purifier $\mathcal{P}$, Retriever $\mathcal{E}$, Max Retrieval Attempts K, Max Revisions M
Output: Answer a

$type \leftarrow \mathcal{R}.\text{AssessComplexity}(Q)$ ▷ Determine if single-hop or multi-hop
if $type =$ `SingleHop` **then**
  $status, info \leftarrow \text{IterativeRetrieval}(Q, \mathcal{R}, \mathcal{P}, \mathcal{E}, K, \emptyset)$ ▷ Call Alg. 2
  $a \leftarrow \mathcal{R}.\text{GenerateAnswer}(Q, info, \emptyset)$
else
  $\mathcal{S} \leftarrow \mathcal{R}.\text{Decompose}(Q)$ ▷ Generate initial sub-question sequence $[q_1, q_2, \ldots, q_n]$
  $\mathcal{C} \leftarrow \emptyset$ ▷ Initialize global context
  $revision_cnt \leftarrow 0$
  $i \leftarrow 1$
  while $i \leq |\mathcal{S}|$ **do**
    $q_i \leftarrow \mathcal{S}[i]$
    $status, info \leftarrow \text{IterativeRetrieval}(q_i, \mathcal{R}, \mathcal{P}, \mathcal{E}, K, \mathcal{C})$ ▷ Call Alg. 2
    if $status =$ `Success` **then**
      $a_i \leftarrow \mathcal{R}.\text{GenerateAnswer}(q_i, info, \mathcal{C})$
      $\mathcal{C} \leftarrow \mathcal{C} \cup \{(q_i, a_i, info)\}$
      $\mathcal{S}[i+1] \leftarrow \mathcal{R}.\text{RefineNextSubQuestion}(\mathcal{S}[i+1], \mathcal{C})$ ▷ Plan Refinement
      $i \leftarrow i + 1$
    else
      if $revision_cnt < M$ **then**
        $\mathcal{S}_{new} \leftarrow \mathcal{R}.\text{RevisePlan}(\mathcal{S}, i, \mathcal{C})$ ▷ Plan Revision
        $\mathcal{S} \leftarrow \mathcal{S}_{new}$
        $revision_cnt \leftarrow revision_cnt + 1$
      else
        $a_i \leftarrow \mathcal{R}.\text{GenerateAnswer}(q_i, \emptyset, \mathcal{C})$ ▷ Answer with internal knowledge
        $\mathcal{C} \leftarrow \mathcal{C} \cup \{(q_i, a_i, \emptyset)\}$
        $i \leftarrow i + 1$
  $a \leftarrow \mathcal{R}.\text{GenerateFinalAnswer}(Q, \mathcal{C})$
return a

Algorithm 2 Iterative Retrieval with Purifier

Input: Target q, Reasoner $\mathcal{R}$, Purifier $\mathcal{P}$, Retriever $\mathcal{E}$, Max Attempts K, Context $\mathcal{C}$
Output: Status, Purified Information

$\mathcal{H} \leftarrow \emptyset$ ▷ Local retrieval history
for $k = 1$ to K **do**
  $query_k \leftarrow \mathcal{R}.\text{GenerateQuery}(q, \mathcal{H}, \mathcal{C})$
  $\mathcal{D}_k \leftarrow \mathcal{E}.\text{Retrieve}(query_k)$ ▷ Retrieve candidate documents
  $info_k, relevance_k \leftarrow \mathcal{P}.\text{Purify}(q, \mathcal{D}_k)$ ▷ Purifier judges relevance and extracts info
  $\mathcal{H} \leftarrow \mathcal{H} \cup \{(query_k, info_k, relevance_k)\}$
  if $relevance_k =$ `Relevant` **then**
    return `Success`, $info_k$
return `Failure`, $\mathcal{H}$

3. **Completion**: Once all sub-questions are solved, the model generates the final answer (Table 22).

C Training Details

SFT Cold-Start We use `qwen-max-latest` (Qwen et al., 2025) as the teacher model to synthesize reasoning trajectories from NQ, HotpotQA,

Algorithm 3 Iterative Retrieval Loop

Input: Retrieval Target q, Teacher Model $\mathcal{M}$, Max Attempts K
Output: Status $\mathcal{S}$, Local Context $\mathcal{H}$

$\mathcal{H} \leftarrow \emptyset$
for $k = 1$ to K **do**
  $r_k, \mathbf{q}_k \leftarrow \mathcal{M}.\text{GenerateQuery}(q, \mathcal{H})$
  $\mathcal{D}_k \leftarrow \text{Retrieve}(\mathbf{q}_k)$
  $I_k, J_k \leftarrow \mathcal{M}.\text{Purify}(q, \mathcal{D}_k)$
  $\mathcal{H} \leftarrow \mathcal{H} \cup \{r_k, \mathbf{q}_k, I_k, J_k\}$
  if $J_k = \texttt{Relevant}$ **then**
    return `Success`, $\mathcal{H}$
return `Failure`, $\mathcal{H}$

Algorithm 4 Trajectory Synthesis for Single-hop Questions

Input: Question Q, Teacher Model $\mathcal{M}$, Max Retrieval Attempts K
Output: Trajectory $\mathcal{T}$ or `Failure`

$r_{type} \leftarrow \mathcal{M}.\text{AssessComplexity}(Q)$ ▷ Identify question complexity
$\mathcal{S}, \mathcal{H} \leftarrow \text{RetrievalLoop}(Q, \mathcal{M}, K)$ ▷ Call Alg. 3
if $\mathcal{S} = \texttt{Success}$ **then**
  $r_{final}, a \leftarrow \mathcal{M}.\text{Answer}(Q, \mathcal{H})$ ▷ Generate final answer
  return $\mathcal{T} = \{Q, r_{type}\} \cup \mathcal{H} \cup \{r_{final}, a\}$
else
  return `Failure`

Algorithm 5 Trajectory Synthesis for Multi-hop Questions

Input: Question Q, Teacher Model $\mathcal{M}$, Max Revisions M, Max Retrieval Attempts K
Output: Trajectory $\mathcal{T}$ or `Failure`

$r_{type} \leftarrow \mathcal{M}.\text{AssessComplexity}(Q)$ ▷ Identify question complexity
$\mathcal{P} \leftarrow \mathcal{M}.\text{Decompose}(Q)$ ▷ Generate initial sub-question sequence
$\mathcal{C} \leftarrow \{Q, \mathcal{P}\}$ ▷ Initialize global context
$revision_cnt \leftarrow 0$
$i \leftarrow 1$ ▷ Index of current sub-question
while $i \leq |\mathcal{P}|$ **do**
  $q_i \leftarrow \mathcal{P}[i]$
  $\mathcal{S}, \mathcal{H} \leftarrow \text{RetrievalLoop}(q_i, \mathcal{M}, K)$ ▷ Call Alg. 3
  if $\mathcal{S} = \texttt{Success}$ **then**
    $\mathcal{C} \leftarrow \mathcal{C} \cup \mathcal{H}$ ▷ Accumulate evidence
    $\mathcal{P} \leftarrow \mathcal{M}.\text{RefineNextSubQuestion}(\mathcal{P}, \mathcal{C})$ ▷ Plan Refinement
    $i \leftarrow i + 1$
  else
    if $revision_cnt < M$ **then**
      $\mathcal{P} \leftarrow \mathcal{M}.\text{RevisePlan}(Q, \mathcal{C})$ ▷ Plan Revision
      $revision_cnt \leftarrow revision_cnt + 1$
      ▷ Resume from the first unsolved sub-question in new plan
    else
      return `Failure`
$a \leftarrow \mathcal{M}.\text{GenerateAnswer}(Q, \mathcal{C})$
$\mathcal{C} \leftarrow \mathcal{C} \cup \{a\}$
return $\mathcal{T} = \{r_{type}\} \cup \mathcal{C}$

MuSiQue (2-hop), and MuSiQue (3,4-hop) training sets, collecting 800, 1,600, 800, and 800 trajectories respectively. This yields 4,000 Reasoner samples and 11,255 Purifier samples. SFT is conducted using LLaMA-Factory (Zheng et al., 2024) with a learning rate of 1e-5, batch size of 64, for 3 epochs. We set the maximum number of retrieval attempts to 3 and the maximum number of plan revisions to 1.

RL Training We sample 5,000, 10,000, 10,000, and 4,762 additional instances (non-overlapping with SFT data) from the NQ, HotpotQA, MuSiQue (2-hop), and MuSiQue (3,4-hop) training sets, totaling 29,762 samples. We adopt GRPO (Shao et al., 2024) as the learning algorithm, implemented on verl (Sheng et al., 2025). Training is conducted for 3 epochs with 5 rollouts per sample, using total/mini/micro batch sizes of 256/128/64, a learning rate of 1e-6, $\alpha = \beta = 0.1$, and discount factor $\lambda = 0.5$. We set the maximum number of retrieval attempts to 3. Following common practice (Jin et al., 2025a,b; Song et al., 2025), we mask out the tokens of search tool responses during training to prevent noise in the retrieval content from skewing the learning signal.

D Datasets

We use publicly available QA benchmarks for training and evaluation. Table 7 summarizes the statistics of all datasets used in our experiments.

Dataset	Train	Dev/Test
NQ	79,168	3,610
SimpleQA	-	4,326
HotpotQA	90,447	7,405
2WikiMultihopQA	15,000	12,576
MuSiQue	19,938	2,417
FRAMES	-	824

Table 7: Statistics of datasets used in our experiments. "-" indicates the split is not used or not available.

Data Sources All datasets except SimpleQA and FRAMES are obtained from FlashRAG (Jin et al., 2024), a modular toolkit for retrieval-augmented generation research that provides standardized dataset preprocessing and evaluation pipelines. These datasets are publicly available and do not contain personally identifiable information or offensive content.

E Ablation Study Implementation Details

For the removal of the Dynamic Global Planning Paradigm (*w/o Plan&Adapt*), we do not use the reasoning trajectory synthesis method described in §2.2. Instead, while retaining the dual-agent architecture, we employ the prompt provided in Table 24 to guide the teacher model in rolling out solutions. We then apply rejection sampling to retain only those reasoning trajectories that yield correct answers, which are subsequently used as SFT data. In the RL phase, we optimize using only format and answer rewards, omitting all planning-related reward designs.

For *w/o Adapt*, we disable the plan adaptation mechanism, meaning the model generates an initial plan but lacks the ability to replan or perform iterative retrieval when retrieval results are irrelevant. During the RL phase, we use the format reward, answer reward, and the Initial Planning Reward R_p.

We conduct SFT training using reasoning trajectories synthesized from the same question set.

For *w/o* R_p and *w/o* R_a, we remove the initial planning reward and adaptation reward respectively, while keeping all other components unchanged.

F Format Reward Constraints

The format reward R_f enforces the following constraints to ensure parsable reasoning paths:

1. **Answer Tag Constraint**: The model output must contain exactly one pair of `<answer>` and `</answer>` tags, enclosing the final answer.

2. **Plan Tag Constraint**: For multi-hop questions, the output must contain `<plan>` and `</plan>` tags that enclose the initial decomposition of sub-questions.

3. **Sub-question Identifier Constraint**: Within the plan, each sub-question must be marked with identifiers "`#Q_i: `" (where i is the sub-question index), enabling systematic tracking of the solving process.

4. **Refinement Tag Constraint**: When refining a sub-question after solving its dependencies, the model must use `<Updated_#Q_i>` and `</Updated_#Q_i>` tags to mark the refined version.

Purifier	LasJ	SC	Len	Time
Search-R1	19.78	3.01	2883	-
w/o Purifier	27.18	4.89	5180	432
Qwen2.5-7B-Instruct	26.70	5.06	1724	359
Qwen2.5-32B-Instruct	28.88	4.89	1758	653
Qwen-Max	32.16	4.65	2073	-
Qwen2.5-3B-SFT	27.30	4.70	1979	345
Qwen2.5-7B-SFT	29.49	4.59	1883	381

Table 8: Performance comparison of different Purifier Agents on FRAMES. LasJ: LLM-as-a-Judge (%); SC: Number of search calls; Len: Context length (tokens); Time: Total inference time (seconds). Search-R1 serves as a baseline without the dual-agent architecture.

5. **Revision Tag Constraint**: When triggering plan revision after retrieval failures, the model must use `<Replan>` and `</Replan>` tags to enclose the revised plan.

These constraints ensure that the reasoning trajectory can be automatically parsed via regular expressions to compute the Initial Planning Reward R_p (by extracting the initial plan and counting subquestions) and the Plan Adaptation Reward R_a (by detecting refinement tags, revision tags, and iterative retrieval patterns).

G Comparison of Different Purifier Agents

This section extends the analysis in §4 by comparing different Purifier configurations. We use the same experimental setup described in the main text.

As shown in Table 8, we compare three categories of Purifier configurations: (1) no Purifier baseline, (2) general-purpose instruction-tuned models (Qwen2.5-7B/32B-Instruct and Qwen-Max), and (3) our task-specific fine-tuned models (Qwen2.5-3B/7B-SFT). Search-R1 serves as an external baseline representing the single-agent paradigm.

Task-Specific Fine-Tuning Enables Smaller Models to Serve as Effective Purifiers. Among general-purpose models, larger Purifiers generally yield better performance, with Qwen-Max achieving the highest LasJ score of 32.16%. However, the general-purpose Qwen2.5-7B-Instruct fails to effectively perform the Purifier task, achieving only 26.70% LasJ—even lower than the no-Purifier baseline (27.18%). This indicates that smaller general-purpose models lack the capability to accurately judge relevance and extract key information from noisy retrieval results. In contrast, our fine-tuned Qwen2.5-7B-SFT achieves a LasJ score of 29.49%, which not only recovers from this deficiency but also surpasses Qwen2.5-32B-Instruct (28.88%) despite having only 22% of its parameters. This demonstrates that task-specific fine-tuning is essential for enabling moderate-sized models to serve as effective Purifiers, offering a practical trade-off between performance and computational cost.

Inference Efficiency Details. We deploy the *Purifier* using vLLM's server mode on an 8×A800 node. Deploying an additional 7B *Purifier* incurs approximately 20% additional GPU memory overhead. The general-purpose Qwen2.5-32B-Instruct not only achieves lower accuracy (28.88% vs. 29.49%) but also incurs significantly higher inference time (653s) due to its larger model size. This comparison highlights the value of task-specific fine-tuning: our Qwen2.5-7B-SFT, with only 22% of the parameters, outperforms the 32B general-purpose model in both accuracy and efficiency.

H Case Study

In this section, we present comparative case studies derived from the interaction trajectories of **Search-R1** and our proposed **D^{2}PLAN**. Specifically, Tables 9 and 10 illustrate a scenario where **Search-R1** exhibits E1 (Deficient Search Chain) by generating a flawed query sequence that diverges from the original reasoning objective—instead of querying the discoverer's birthdate, it directly searches for a holiday, leading to an incorrect answer. In contrast, our proposed **D^{2}PLAN** successfully solves the problem through explicit global planning and dynamic subquestion refinement, maintaining alignment with the original reasoning objective throughout the process.

Additionally, Tables 11 and 12 demonstrate how **D^{2}PLAN**'s plan revision mechanism corrects an initially flawed reasoning direction. In this example, **Search-R1** incorrectly identifies "line" as a synonym for "leash" and then erroneously concludes that "Lithium" sounds like "line", leading to an incorrect answer. In contrast, **D^{2}PLAN**'s initial decomposition leads to semantic drift—retrieving engineering "dog" tools instead of pet walking equipment. After two failed retrieval attempts, the model triggers plan revision that merges the first two subquestions, directly querying for synonyms of dog walking tools. This successfully redirects the reasoning and arrives at the correct answer "Lead"

Question: The Basibasy mine is located in Madagascar. This mine is abundant in a specific chemical element that was discovered for the first time in 1791. The person who discovered this element was born on what is now known as a major US holiday - what holiday is this?
Golden Answer: Christmas

Search-R1 (Incorrect):
<search> Basibasy mine Madagascar </search>
<information> Doc 1(Title: "Ambatovy mine") Ambatovy, located in Madagascar, off the east coast of Africa, is one of the world's largest lateritic nickel mines with a production capacity of 60,000 tonnes of nickel... Doc 2(Title: Maniry) Maniry is a town and commune in southwestern Madagascar. It belongs to the district of Ampanihy... Doc 3(Title: "Economy of Madagascar") coal at Sakoa and nickel near Tamatave. The Ambatovy nickel mine is a huge operation... Doc 4(Title: "Ambatovy mine") Bank CSRM report estimate the impact of the mining industry as two to five indirect jobs... Doc 5(Title: "Mining industry of Madagascar") of mining Code. Minerals, both industrial and metallic, are numerous including ilmenite, graphite... </information>
Based on the information provided, the Basibasy mine seems to be referred to as Ambatovy mine, which is located in Madagascar and is a large nickel mine. Now, let's find the element discovered in 1791.
<search> element discovered in 1791 </search>
<information> Doc 1(Title: "Group 9 element") regia. Iridium was discovered in a very similar way in 1804 by Smithson Tennant... Doc 2(Title: Titanium) Titanium was discovered in 1791 by the clergyman and amateur geologist, William Gregor, as an inclusion of a mineral in Cornwall, Great Britain... Doc 3(Title: "Group 3 element") the current heaviest element known, oganesson. In 1787, Swedish part-time chemist Carl Axel Arrhenius... Doc 4(Title: "Anders Gustaf Ekeberg") Anders Gustaf Ekeberg was a Swedish chemist who discovered tantalum in 1802... Doc 5(Title: "Group 9 element") Meitnerium has not been isolated in pure form... </information>
From the information provided, it seems William Gregor discovered the element titanium in 1791. Now, let's find the US holiday related to the birth date provided.
<search> person born on Thanksgiving </search>
<information> Doc 1(Title: "Thanksgiving (United States)") Thanksgiving is a national holiday in the United States, celebrated on the fourth Thursday of November. It originated as a harvest festival... Doc 2(Title: Thanksgiving) Thanksgiving Day is a national holiday celebrated on various dates in Canada, the United States... Doc 3(Title: Thanksgiving) after 1700, late autumn Thanksgiving celebrations became commonplace... Doc 4(Title: Thanksgiving) beginning of the 19th century, coinciding with and eventually superseding the existing holiday... Doc 5(Title: "Preterm birth") Ahmad resuscitated the baby after she was born... </information>
The information states that Thanksgiving is a national holiday in the United States, celebrated on the fourth Thursday of November. The person who discovered titanium, William Gregor, was born on a day that is now a US holiday. Therefore, the holiday in question is likely Thanksgiving.
<answer> Thanksgiving </answer>

Table 9: Case study: Search-R1 exhibits E1 (Deficient Search Chain) by generating a flawed query sequence. Instead of querying William Gregor's birthdate, it directly searches for "person born on Thanksgiving", diverging from the correct reasoning objective and leading to an incorrect answer.

Question: The Basibasy mine is located in Madagascar. This mine is abundant in a specific chemical element that was discovered for the first time in 1791. The person who discovered this element was born on what is now known as a major US holiday - what holiday is this?
Golden Answer: Christmas

D^{2}PLAN (Correct):
This is a multi-hop question involving multiple steps: identifying the chemical element discovered in 1791 at the Basibasy mine in Madagascar, determining the discoverer's birth date, and linking it to a significant U.S. holiday. Each step builds on the previous one, requiring sequential reasoning.
<plan>
#Q_1: What chemical element was discovered for the first time in 1791 and is abundant in the Basibasy mine in Madagascar?
#Q_2: On what day was the person who discovered the element identified in #A_1 born?
#Q_3: What major US holiday is associated with the birthdate identified in #A_2?
</plan>
To proceed, I'll first identify the chemical element linked to both the Basibasy mine and its discovery in 1791. External verification will confirm whether this aligns with historical records of discoveries and elemental abundances.
<tool_call>
{"name": "search", "arguments": {"query": "chemical element discovered in 1791 in Madagascar", "question": "What chemical element was discovered for the first time in 1791 and is abundant in the Basibasy mine in Madagascar?"}}
</tool_call>
<tool_response>
{"result": "Doc 1: [TARGET_INFO_EXTRACTED] Titanium was discovered in 1791 by the clergyman and amateur geologist William Gregor; Titanium is abundant in the Basibasy mine in Madagascar"}
</tool_response>
The document states that titanium was discovered in 1791 and is abundant in the Basibasy mine in Madagascar, directly answering the question.
#A_1: Titanium
I'll now determine the discoverer's birthdate:
<updated_#Q_2>
On what day was William Gregor, the person who discovered titanium, born?
</updated_#Q_2>
The prior answer identifies titanium but does not include details about William Gregor's birthdate. External information is needed to address this sub-question.
<tool_call>
{"name": "search", "arguments": {"query": "William Gregor birth date", "question": "On what day was William Gregor, the person who discovered titanium, born?"}}
</tool_call>
<tool_response>
{"result": "Doc 1: [TARGET_INFO_EXTRACTED] William Gregor was born on 25 December 1761"}
</tool_response>
The document provides William Gregor's birth date, including the day.
#A_2: 25 December
I'll now determine which major U.S. holiday corresponds to this date.
<updated_#Q_3>
What major US holiday is celebrated on December 25?
</updated_#Q_3>
The answer is common knowledge: Christmas Day, a widely recognized national holiday in the United States.
#A_3: Christmas Day
<answer>
Christmas Day
</answer>

Table 10: Case study: D^{2}PLAN succeeds through structured planning and dynamic sub-question refinement, effectively mitigating E1 (Deficient Search Chain). By explicitly decomposing the problem and dynamically updating #Q_2 with the discovered entity (William Gregor), it avoids diverging from the original reasoning objective and correctly queries the birthdate, arriving at the correct answer.

(both a synonym for leash and a chemical element).

I Error Analysis: E2.1 vs. E2.2 Error Case

Tables 13 and 14 illustrate the difference between E2.1 and E2.2 errors, comparing the behaviors of Search-R1 and D^{2}PLAN on the same question.

As shown in Table 13, Search-R1 successfully retrieves Hod Lisenbee's place of death (Clarksville) but fails to find the distance information in the second retrieval. Critically, **Search-R1 does not attempt to reformulate the query or re-retrieve**—instead, it directly hallucinates an answer ("approximately 60 miles") based on its internal knowledge, which is incorrect. This exemplifies **E2.1 errors**: the model reasons with irrelevant retrieval results without attempting any remediation.

In contrast, as shown in Table 14, D^{2}PLAN demonstrates more robust behavior: when the initial retrieval for the distance fails, the model appropriately recognizes the irrelevance and attempts query reformulation with enriched entities (adding constraints like "miles", "km", and authoritative sources). However, despite multiple retrieval attempts with refined queries, the retrieval system cannot locate the required distance information, causing the model to hallucinate an incorrect answer (95 miles instead of 45 miles). This exemplifies **E2.2 errors**: the model attempts remediation but fails due to inherent corpus limitations.

Upon manual inspection, we traced the ground-truth answer to the Wikipedia article for Clarksville, Tennessee (https://en.wikipedia.org/wiki/Clarksville,_Tennessee), which states: "Clarksville is located on the northwest edge of the Highland Rim, which surrounds the Nashville Basin, and is 45 miles (72 km) northwest of Nashville." However, the corresponding document in our corpus does not contain the critical distance information "45 miles (72 km)":

```
{"id": "14985934", "contents": ""Clarksville, Tennessee" was rededicated. In addition to restoring the 1879 courthouse and plazas, the county built a new courts center on the north side for the court operations. According to the United States Census Bureau, the city has a total area of , of which is land and (0.71%) is covered by water. Clarksville is located on the northwest edge of the Highland Rim, which surrounds the Nashville Basin, and is northwest of Nashville. Fort Campbell North is a census-designated place (CDP) in Christian County, Kentucky. It contains most of the housing for the Fort Campbell Army base. The population was 14,338"}
```

As shown, the corpus document retains only the directional phrase "northwest of Nashville" without the numeric distance, making it impossible for the model to retrieve the correct answer. This comparison highlights the key difference between E2.1 and E2.2 errors: Search-R1 exhibits E2.1 behavior by immediately resorting to hallucination without any remediation effort, whereas D^{2}PLAN demonstrates superior self-correction behavior by actively attempting to reformulate queries—though it ultimately fails due to corpus limitations (E2.2). This case study illustrates how D^{2}PLAN's dynamic planning paradigm shifts errors from model capability deficiencies (E2.1) to inherent retrieval system limitations (E2.2).

J Prompts

Question: Which element of the periodic table is a homonym of a synonym of a tool commonly used in dog walking? **Golden Answer**: Lead
Search-R1 (Incorrect): To solve this problem, I'll need to break it down step by step: 1. Identify a tool commonly used in dog walking. 2. Find a synonym for that tool. 3. Find an element in the periodic table that is a homonym of that synonym. Step 1: A tool commonly used in dog walking is a leash or a lead. Step 2: A synonym for "leash" or "lead" could be "line." Step 3: I need to find an element in the periodic table that is a homonym of "line." <search> homonym of line periodic table </search> <information> Doc 1(Title: "Homology (chemistry)") Homology (chemistry) In chemistry, homology is the appearance of homologues. A homologue is a compound belonging to a series of compounds differing from each other by a repeating unit... Doc 2(Title: "Periodic table") The periodic table is a tabular arrangement of the chemical elements, ordered by their atomic number... </information> From the information provided, I can see that there is no direct mention of an element that is a homonym of "line." However, the periodic table contains elements with various names. I will search for elements with names that sound like "line." <search> elements periodic table names sound like line </search> <information> Doc 1(Title: "Periodic table") The periodic table is a tabular arrangement of the chemical elements... Doc 2(Title: "Naming of chemical elements") Chemical elements may be named from various sources: sometimes based on the person who discovered it, or the place it was discovered... </information> There is no element in the periodic table that is a homonym of "line." I will try searching for elements with names that sound like "line." <search> elements periodic table names sound like line or lean </search> <information> Doc 1(Title: "Periodic table") The periodic table is a tabular arrangement of the chemical elements... Doc 2(Title: "Chemical element") The terms "light" and "heavy" are sometimes also used informally to indicate relative atomic numbers... </information> After searching, I found that there is an element in the periodic table named "Lithium," which sounds like "line." Therefore, the answer is: <answer> Lithium </answer>

Table 11: Case study: Search-R1 incorrectly identifies "line" as a synonym for "leash" and then erroneously concludes that "Lithium" sounds like "line." The model fails to recognize that "lead" is both a synonym for leash and a chemical element, leading to an incorrect answer.

Question: Which element of the periodic table is a homonym of a synonym of a tool commonly used in dog walking?
Golden Answer: Lead

D^{2}PLAN (Correct):
This is a multi-hop question involving two steps: identifying a tool used in dog walking and finding an element whose name matches a synonym of that tool. The phrasing introduces ambiguity and requires sequential reasoning.
<plan>
#Q_1: What is a tool commonly used in dog walking?
#Q_2: What is a synonym of the tool identified in #A_1?
#Q_3: Which element of the periodic table is a homonym of the synonym identified in #A_2?
</plan>
First, I'll identify a tool frequently used in dog walking. External knowledge is needed since no prior context provides this information.
<tool_call>
{"name": "search", "arguments": {"query": "common tool used in dog walking", "question": "What is a tool commonly used in dog walking?"}}
</tool_call>
<tool_response>
{"result": "[NO_TARGET_INFO_FOUND]\nSummary of retrieved documents:\n[Doc 1]: Dog walking involves a person walking with a dog... [Doc 2]: Dog walking refers to the act of walking a dog... [Doc 3]: In engineering, a dog is a tool or part... [Doc 4]: Engineering tools called 'dogs' include feed dogs... [Doc 5]: A bench dog is a woodworking accessory..."}
</tool_response>
The question asks about a tool used in dog walking. However, the documents describe 'dog' as an engineering term referring to tools like dog clutches, feed dogs in sewing machines, and bench dogs in woodworking—not as a pet-related term. The documents lack sufficient semantic alignment to answer the question.
Let me modify my search query using Entity Enrichment / Correction strategy.
<tool_call>
{"name": "search", "arguments": {"query": "engineering tool named dog — dog clutch, feed dog in sewing machine, or bench dog in woodworking", "question": "What is a tool commonly used in dog walking?"}}
</tool_call>
<tool_response>
{"result": "[NO_TARGET_INFO_FOUND]\nSummary of retrieved documents:\n[Doc 1]: In engineering, a dog is a tool or part... [Doc 2]: A dog in engineering refers to a tool or mechanism... [Doc 3]: A lathe dog, also known as a lathe carrier... [Doc 4]: A bench dog is a woodworking accessory... [Doc 5]: Shutter dogs are wrought-iron or cast-iron devices..."}
</tool_response>
The documents still describe 'dog' in engineering contexts but do not mention dog walking or related activities. (No relevant docs found after 2 attempts.)
Plan revision triggered due to repeated irrelevant retrieval.
<replan>
#Q_1: What is a synonym of a tool commonly used in dog walking?
#Q_2: Which element of the periodic table is a homonym of the synonym identified in #A_1?
</replan>
The first step is addressing #Q_1. External information is required since the context does not specify a tool for dog walking or its synonyms.
<tool_call>
{"name": "search", "arguments": {"query": "What is a synonym for leash or harness, a tool commonly used in dog walking?", "question": "What is a synonym of a tool commonly used in dog walking?"}}
</tool_call>
<tool_response>
{"result": "Doc 1: [TARGET_INFO_EXTRACTED] A leash is also called a lead, lead line or tether"}
</tool_response>
The document states that a leash is also called a lead, lead line, or tether—synonyms for a tool used in dog walking.
#A_1: Lead
I'll now determine which chemical element shares its name with this synonym.
<updated_#Q_2>
Which element of the periodic table is a homonym of the synonym "lead"?
</updated_#Q_2>
The synonym identified was 'lead', which is already a chemical element (Pb). No external retrieval is necessary.
#A_2: Lead
<answer>
Lead
</answer>

Table 12: Case study: D^{2}PLAN's plan revision mechanism corrects semantic drift through dynamic replanning.

Question: How far is Hod Lisenbee's place of death in TN from Nashville?
Golden Answer: 45

Search-R1 (Incorrect — E2.1 Error):
<think> I need to find out the place of death for Hod Lisenbee first. </think>
<search> place of death for Hod Lisenbee </search>
<information> Doc 1(Title: "Hod Lisenbee") He died at age 89 on November 14, 1987, in Clarksville and is buried at the Liberty Presbyterian Church Cemetery. In 1998, the Clarksville City Council renamed a portion of Dover Road (Highway 79) near the Dover Crossing interception as Hod Lisenbee memorial highway... Doc 2(Title: "Hod Lisenbee") retired from baseball. Hod Lisenbee was manager and half-owner of the Clarksville Colts club from 1946 to 1948... Lisenbee lived in his hometown of Clarksville from the fall of 1945, until his death in 1987... Doc 3(Title: "Hod Lisenbee") from the Cincinnati Reds and pitched 31 games for the Reds, as a reliever... Following his release from the Reds, he returned to his hometown of Clarksville... </information>
From the information, Hod Lisenbee's place of death is in Clarksville. Now I need to find out how far Clarksville is from Nashville.
<search> distance from Clarksville to Nashville </search>
<information> Doc 1(Title: "Clarksville, Tennessee") Clarksville, Tennessee Clarksville is the county seat of Montgomery County, Tennessee, United States. It is the fifth-largest city in the state behind Nashville, Memphis, Knoxville, and Chattanooga. The city had a population of 132,929 at the 2010 census... Doc 2(Title: "Nashville Metropolitan Transit Authority") public hearings on April 24, 2012 regarding proposed regional express bus service from Clarksville to Nashville... Doc 3(Title: "Clarksville, Tennessee") Pleasant Academy. By 1819, the newly established town had 22 stores... Trade via land also grew as four main dirt roads were established, two to Nashville... </information>
From the information, Clarksville is the fifth-largest city in Tennessee, and it is located in Montgomery County. Nashville is the capital of Tennessee and the county seat of Davidson County. The distance from Clarksville to Nashville is approximately 60 miles.
<answer> 60 miles </answer>

Table 13: Case study illustrating **E2.1 error** of Search-R1: the model fails to retrieve the distance information and **does not attempt to reformulate the query**. Instead, it directly hallucinates an incorrect answer (60 miles) based on internal knowledge, demonstrating lack of self-correction capability.

Question: How far is Hod Lisenbee's place of death in TN from Nashville?
Golden Answer: 45

D^{2}PLAN (Incorrect — E2.2 Error):
This is a multi-hop question requiring two steps: identifying Hod Lisenbee's place of death in Tennessee and calculating its distance from Nashville. Each step depends on resolving the prior one.
<plan>
#Q_1: Where did Hod Lisenbee die?
#Q_2: How far is #A_1 (Hod Lisenbee's place of death) from Nashville, TN?
</plan>
I'll first determine where Hod Lisenbee died. This information isn't provided in the question or prior context, so it must be retrieved externally.
<tool_call>
{"name": "search", "arguments": {"query": "Where did Hod Lisenbee die?", "question": "Where did Hod Lisenbee die?"}}
</tool_call>
<tool_response>
{"result": "Doc 1: [TARGET_INFO_EXTRACTED] Hod Lisenbee died in Clarksville"}
</tool_response>
The document states that Hod Lisenbee died in Clarksville.
#A_1: Clarksville
With Clarksville identified as the location of Hod Lisenbee's death, I'll now calculate the distance from there to Nashville, TN.
<updated_#Q_2>
How far is Clarksville, TN from Nashville, TN?
</updated_#Q_2>
This requires determining the geographical distance between the two cities, which hasn't been covered yet.
<tool_call>
{"name": "search", "arguments": {"query": "distance from Clarksville TN to Nashville TN", "question": "How far is Clarksville, TN from Nashville, TN?"}}
</tool_call>
<tool_response>
{"result": "[NO_TARGET_INFO_FOUND] The Clarksville, TN–KY metropolitan statistical area consists of Montgomery and Stewart Counties in Tennessee, and Christian and Trigg Counties in Kentucky; The city distance between Clarksville, TN from Nashville is not provided..."}
</tool_response>
The document mentions the Clarksville, TN–KY metropolitan statistical area but does not provide the actual distance. Thus, it lacks sufficient information to answer the question directly.
The retrieved documents are not relevant. I'll modify my search query. The previous query returned contextual details about the Clarksville-Nashville region but failed to yield the numeric distance due to missing explicit entity constraints like units ("miles", "km") and authoritative source signals.
<tool_call>
{"name": "search", "arguments": {"query": "What is the straight-line driving distance in miles between Clarksville, Tennessee and Nashville, Tennessee, according to the U.S. Census Bureau or Tennessee Department of Transportation?", "question": "How far is Clarksville, TN from Nashville, TN?"}}
</tool_call>
<tool_response>
{"result": "[NO_TARGET_INFO_FOUND] The Clarksville, TN–KY metropolitan statistical area consists of Montgomery and Stewart Counties in Tennessee...; The public hearings on April 24, 2012 regarding proposed regional express bus service from Clarksville to Nashville indicate that the distance between Clarksville and Nashville is significant enough to require express bus service."}
</tool_response>
The document mentions that public hearings discussed proposed regional express bus service from Clarksville to Nashville, noting the distance is significant. However, it does not provide the exact numerical distance or any quantitative measure (e.g., miles or kilometers), only qualitative information about its significance.
#A_2: Approximately 95 miles
<answer>
95 miles
</answer>

Table 14: Case study illustrating E2.2 error of D^{2}PLAN.

Question Decomposition Prompt

I will give you a multi-hop question. You need to decompose it into atomic sub-questions. Since the sub-questions may depend on each other, you need to number the sub-questions (#Q_1, #Q_2, #Q_3...). A sub-question numbered k can only depend on all sub-questions numbered less than k. When considering dependencies, some sub-questions may rely on the answers to other sub-questions. In this case, you need to denote these answers as #A_i, where i is the corresponding sub-question number.

Please follow these instructions.
- Be creative and varied in your response.
- You should output in the `<output></output>` tags
- First provide the rationale for the decomposition, then list the sub-questions.
- The reasonable number of sub-questions is between 2 and 5.
- Do not output subtitles like "Rationale for the Decomposition:", but directly provide the rationale.
- You need to place the subquestions within the `<plan></plan>` tag.

An Example of a decomposition:

```
<question>
Here is a multi-hop question
</question>

<output>
Here is the rationale for the decomposition

<plan>
#Q_1: Sub-question 1
#Q_2: Sub-question 2
...
</plan>
</output>
```

Question: {question}

Table 15: Prompt template for question decomposition in multi-hop QA.

Query Rewrite Prompt

You are an expert search strategist for a semantic retrieval system. Your goal is to analyze and refine a search query that has previously failed to retrieve relevant documents.

Some common rewriting strategies include: expanding the query into a hypothetical answer, breaking down the question into sub-questions, adding or correcting key entities, asking a more general high-level question, or narrowing the query with specific constraints. You may use these or any other strategy you find appropriate.

Based on the user's question and the failed search history, your task is to generate a new, improved query. Provide your output as a single JSON object with no other text before or after it.

Original question: {question}

Previous search attempts (all failed to retrieve relevant information): {query_history}

JSON Output:
You must output a JSON formatted object that includes:
- "reason": A concise analysis of why the previous queries likely failed and a justification for your new approach.
- "new_query": The modified search query, designed for better semantic retrieval.

Table 16: Prompt template for query rewriting in iterative retrieval.

Purifier Agent Prompt

TASK: Analyze the provided [DOCUMENTS] to determine whether they contain information that answers the [QUESTION]. Then output EXACTLY ONE valid JSON object following the [RULES] and [JSON OUTPUT FORMAT].

RULES:

1. RELEVANT DETERMINATION:
   If ANY document provides information that can help to answer the QUESTION, set "relevant" to "Yes"; otherwise "No".

2. WHEN relevant = "Yes":
   - "extracted_info": Output ONLY the minimal factual statements (facts) present in the documents that are strictly required to answer the QUESTION. Do NOT output a full formulated answer, do NOT repeat or restate the QUESTION.
   - Each fact must be directly supported by the documents; paraphrasing is allowed but no speculation or inference beyond what is stated.
   - Each fact MUST be SELF-CONTAINED: explicitly name the subject/entity; do NOT use context-dependent pronouns or vague referents (e.g. it, they, he, she, this, that, these, those, its, their, former, latter). Replace such pronouns with the concrete entity name from the documents.
   - If multiple facts are needed, separate them with "; " (semicolon + space). If one fact suffices, output just that fact.
   - No document indices, no citations, no commentary, no prefixes like "Fact:" or "According to".
   - "summary": must be an empty string "".

3. WHEN relevant = "No":
   - "extracted_info": must be "".
   - "summary": Provide brief summaries for EACH document using the exact format:
     [Doc 1]: <summary of document 1>\n[Doc 2]: <summary of document 2>\n...
   Each summary should be concise and factual.

4. OUTPUT:
   - Return ONLY the JSON object; no extra text, no markdown.
   - Ensure valid JSON (double quotes, proper escaping).

QUESTION: {query}

DOCUMENTS:
{docs_text}

JSON OUTPUT FORMAT (fields and meaning):
"relevant": "Yes" or "No"
"extracted_info": See RULES
"summary": See RULES

OUTPUT:

Table 17: Prompt template for the Purifier Agent to extract relevant information from retrieved documents.

Sub-question Answering Prompt

You need to answer the question. Please concisely analyze it first before providing your answer. You should provide your answer at the end in the `<answer></answer>` tag.

Rules:
- If you think that there is no information in the document to answer the question, please first indicate that there is no information, and then use your internal knowledge to answer the question.
- Do NOT include any markdown formatting in your response like "# Answer" or "## Explanation".
- The final answer in the `<answer></answer>` tag should be concise (no more than 15 words) and directly address the original question. It can be a single word or a phrase without markdown formatting.

```
<document>
{document}
</document>
<question>
{question}
</question>
```

Table 18: Prompt template for answering sub-questions with retrieved documents.

Sub-question Update Prompt

I will give you a multi-hop question and several sub-questions decomposed from it. Sub-questions 1∼{k} have been solved. Now please update sub-question {j} based on the questions and answers of sub-questions 1∼{k}, making it self-contained. Only output the updated sub-question.

Multi-hop question:
{question}

Sub-questions and answers:
{sub_qa}

Updated #Q_{j}:

Table 19: Prompt template for updating sub-questions based on solved dependencies.

Retrieval Necessity Judgment Prompt

I will give you the original question, previously solved sub-questions and their answers, and the current sub-question to be solved. Please determine whether you need to retrieve external information to answer the current sub-question based on the information already available.

You need to output a JSON formatted object that includes:
- "reason": a concise reason for whether you need to retrieve external information
- "need_retrieval": true/false, indicating whether retrieval is needed
- "query": if need_retrieval is true, please provide the query for retrieval; otherwise, set to null

Original question: {original_question}

Previously solved sub-questions and answers:
{previous_qas}

Current sub-question to solve: {current_subquestion}

Table 20: Prompt template for judging whether retrieval is needed for a sub-question.

Plan Revision Prompt

I will give you the original multi-hop question and the solved sub-questions with their answers so far. You need to REVISE the remaining plan by generating a new ordered list of self-contained replacement sub-questions that efficiently lead to the final answer.

Please follow these instructions.
- Do not repeat already-solved sub-questions.
- Each new sub-question must be self-contained and should implicitly leverage the known answers.
- Keep the list concise (between 1 and 5 items).
- A sub-question numbered k can only depend on all sub-questions numbered less than k. When considering dependencies, some sub-questions may rely on the answers to other sub-questions. In this case, you need to denote these answers as #A_i, where i is the corresponding sub-question number.
- Output strictly within the `<output></output>` tags.
- Inside `<output>`, write only lines in the following format, one per line:
  #Q_1: <sub-question>
  #Q_2: <sub-question>
  ...

An Example:

```
<output>
#Q_1: xxxx
#Q_2: xxxx based on #A_1
#Q_3: xxxx based on #A_1 and #A_2
</output>
```

Original multi-hop question:
{question}

Solved sub-questions and their answers so far:
{solved_qas}

Table 21: Prompt template for plan revision after retrieval failures.

Final Answer Synthesis Prompt
Based on the analysis of the multi-hop question and the sub-questions, please provide a final answer that synthesizes the information from all relevant sub-questions. Only output the final answer. The final answer should be concise (no more than 10 words) and directly address the original multi-hop question. It can be a single word or a phrase without markdown formatting. **Multi-hop question:** {question} **Sub-questions and answers:** {sub_qa} **Final answer:**

Table 22: Prompt template for synthesizing the final answer from sub-question answers.

LLM-as-a-Judge Prompt Template
You are a judge. You need to determine whether the answer to a question provided by the LLM is correct based on the reference answer. Please answer "Yes" if they are consistent and "No" if they are not. Only output "Yes" or "No". **Question:** {question} **LLM answer:** {llm_answer} **Reference answer:** {reference_answer}

Table 23: Prompt template for LLM-as-a-Judge (LasJ) evaluation.

Reasoner Agent Prompt (w/o Global Planning)
I will give you a question. You can call a retriever to obtain relevant documents to help answer the question, or you can directly use internal knowledge to respond. If you think there is no relevant information in the search results, you can call the retriever again. You can call the retriever multiple times in the multi-turn conversation. You must conduct reasoning inside `<think>` and `</think>` first every time you call the retriever. You should think and analyze it first, and then provide the answer at the end in the `<answer></answer>` tags. The answer in the `<answer></answer>` tag should be concise (no more than 15 words) and directly address the original question. **Question:** {question}

Table 24: Prompt template for the Reasoner Agent without global planning mechanism, used in the ablation study (*w/o Global Planning*).

Trajectory Rewriting Prompt for $\mathcal{A}_r$

You are an expert at transforming structured reasoning trajectories into coherent, natural reasoning processes. Your task is to rewrite the given trajectory into a fluent and logical reasoning flow that can serve as a high-quality training example for a Reasoner Agent.

Input Format
You will receive a structured reasoning trajectory.
Output Requirements
Transform the trajectory into a natural reasoning process that:
1. Maintains all the key reasoning steps and information from the original trajectory
2. Uses a coherent narrative flow with smooth transitions between steps
3. Preserves the structured tags required for training:
   - `<tool_call>...</tool_call>` for each retrieval attempt
   - `<tool_response>...</tool_response>` for each retrieval result
   - `<plan>...</plan>` for initial question decomposition
   - `<Updated_#Q_i>...</Updated_#Q_i>` for sub-question updates
   - `<Replan>...</Replan>` for global replanning
   - `<answer>...</answer>` for the final answer
4. Removes redundant or verbose content while keeping essential reasoning
5. Ensures the reasoning appears natural and self-contained

Guidelines
- The rewritten trajectory should read as if a skilled reasoner is thinking through the problem step by step
- Integrate the retrieval feedback naturally into the reasoning flow
- When sub-questions are updated, explain the reasoning behind the update
- When replanning occurs, articulate why the previous approach failed and how the new plan addresses it
- Keep the final answer concise and directly addressing the original question

Original Trajectory:
{trajectory}

Rewritten Trajectory:

Table 25: Prompt template for rewriting synthesized trajectories into coherent reasoning processes for Reasoner's SFT training (§2.2.2).

Error Analysis Prompt

You are a professional Search Agent Error Diagnostic Expert. Your task is to analyze the reasoning trajectory of an Agent in Open-domain Multi-hop QA tasks and classify errors based on the criteria provided below.

Please strictly follow the "First Error" principle: check the reasoning trajectory chronologically from the beginning. Once you identify an error fitting any of the following categories, immediately stop checking subsequent steps and output the result.

Type 1: Erroneous Question Decomposition
The Agent makes a mistake during question decomposition, resulting in an incorrect sequence of sub-questions/sub-queries. Examples include:
- The generated series of sub-questions deviates from the original question or covers insufficient knowledge to effectively support the answer to the complex task.
- The decomposed sub-questions are not atomic enough or are too granular.
- When facing a factual sub-question, the Agent fails to invoke a search tool and directly answers incorrectly based on internal parametric knowledge.

Type 2: Susceptibility to Contextual Noise (Direct Reasoning)
The model directly reasons with irrelevant retrieval results without attempting remediation. Specifically, the planning logic and the sub-question itself are correct, but the retrieval results consist entirely of irrelevant information. The Agent makes no attempt to remedy the situation by modifying the Query or re-decomposing the sub-questions; instead, it answers directly based on the irrelevant information or incorrect internal knowledge.

Type 3: Susceptibility to Contextual Noise (Failed Remediation)
The model recognizes that the initial retrieval is irrelevant and attempts query rewriting several times but still fails to obtain correct evidence, resulting in an incorrect answer. Specifically, the planning logic and the sub-question itself are correct. The Agent initiates a search, but the retrieval results consist entirely of irrelevant information. The Agent attempts remedial measures (e.g., rewriting the Query, modifying the sub-question decomposition), but the retrieval results still do not contain relevant information. Consequently, the Agent answers incorrectly due to a lack of information or is forced to answer "not found."

Type 4: Other Errors
Errors that do not belong to any of the above categories.

Input Data
[Original Question]
{question}
[Reasoning Trajectory]
{trajectory}
[Golden Answer]
{golden_answer}

Output Format
Please output the analysis result in JSON format:
```
{
  "analysis": "A brief analysis of the reasoning trajectory, explaining why it is classified into this error type.",
  "error_type": "Type 1 / Type 2 / Type 3 / Type 4",
}
```

Table 26: Prompt template for error analysis using `gemini-3-pro-preview`.

Redundancy Check Prompt

You are a question decomposition expert. Please determine whether the model-generated question decomposition plan contains **redundant subquestions**.

Definition of Redundant Subquestions:
- The subquestion does **NOT help** solve the original question; it is unnecessary
- The subquestion has **NO semantic correspondence** in the Gold-annotated question decomposition
- Even if the wording differs, as long as they ask about the same thing, they are considered semantically corresponding

Example of Redundant Question:
- Original question: "Where is Ulrich Walter's employer headquartered?"
- Gold decomposition: Q1: Ulrich Walter >> employer, Q2: #1 >> headquarters location (2 steps)
- Model decomposition: Q1: Who is Ulrich Walter?, Q2: Who is Ulrich Walter's employer?, Q3: Where is #A_2 headquartered? (3 steps)
- "Who is Ulrich Walter?" is **redundant** because it does not help solve the original question, and there is no corresponding question in Gold

Example of Non-Redundant Question:
- Original question: "Who is the spouse of the Green performer?"
- Gold decomposition: Q1: Green >> performer, Q2: #1 >> spouse (2 steps)
- Model decomposition: Q1: Who is the Green performer?, Q2: Who is the spouse of #A_1? (2 steps)
- "Who is the Green performer?" is **NOT redundant** because it is equivalent to Gold's Q1 "Green >> performer" and is a necessary step to solve the original question

Notes:
- "#1" in Gold refers to the answer of the first question; "#A_1" in the model means the same thing
- "X >> Y" format means asking for the Y attribute of X, equivalent to "What/Who is the Y of X?"
- As long as the model's subquestion **semantically corresponds** to a question in Gold, it is NOT redundant
- If the model's subquestion has no semantic correspondence in Gold AND does not help solve the original question, it is redundant

Original Question: {original_question}

Gold Decomposition ({gold_steps} steps):
{gold_str}

Model Decomposition Plan ({model_steps} steps):
{model_str}

Please carefully analyze each subquestion from the model:
1. Does this subquestion have a semantically corresponding question in Gold?
2. If not, does this subquestion help solve the original question?
3. If there is no correspondence AND it does not help, then the question is redundant

Please respond in the following JSON format (field names must match exactly):
```
{
  "has_extra": true,
  "extra_questions": ["Q1: Who is Ulrich Walter?"],
  "explanation": "Q1 does not help solve the original question and has no correspondence in Gold"
}
```
Or if there are no redundant questions:
```
{
  "has_extra": false,
  "extra_questions": [],
  "explanation": "All questions have correspondence or help solve the original question"
}
```

Output only JSON, nothing else.

Table 27: Prompt for detecting redundant sub-questions using `gemini-3-flash-preview`.